CrossTalk Series

An excellent resource for building security in is *CrossTalk: The Journal of Defense Software Engineering*[1], co-sponsored by DHS.

Problems have been reported when reading these files in Firefox 3 using the Adobe Reader in Firefox.

The DHS NCSD co-sponsored issues include the following:

• March 2007: Software Security[2]

• May 2007: Software Acquisition[3]

• September 2007: Service Oriented Architecture[4]

• June 2008: Software Quality[5]

•September 2008: Application Security[6]

• Mar/Apr 2009: Reinforcing Good Practices[7]

• Sep/Oct 2009: Resilient Software[8]

• Mar/Apr 2010: Systems Assurance[9]

• Sep/Oct 2010: Software Assurance[10]

All the articles in the following issues are closely related to building security in:

Sep/Oct 2009: Resilient Software[11]

Mar/Apr 2010: Systems Assurance[12]

Sep/Oct 2010: Software Assurance[13]

Although there are many excellent CrossTalk articles, we find that the following articles are particularly relevant to the work of the Build Security In website:

Name	Publication Date	Abstract
Baking in Security During the Systems Development Life Cycle	3/1/07	Kwok H. Cheng. Vulnerabilities in software that are introduced by mistake or poor practices pose a serious threat. Combating threats in today's electronic environment requires a methodical approach in building security into software from the ground up, or baking in security as some

1. http://www.crosstalkonline.org/
2. http://www.crosstalkonline.org/storage/issue-archives/2007/200703/200703-0-Issue.pdf
3. http://www.crosstalkonline.org/storage/issue-archives/2007/200705/200705-0-Issue.pdf
4. http://www.crosstalkonline.org/storage/issue-archives/2007/200709/200709-0-Issue.pdf
5. http://www.crosstalkonline.org/storage/issue-archives/2008/200806/200806-0-Issue.pdf
6. http://www.crosstalkonline.org/storage/issue-archives/2008/200809/200809-0-Issue.pdf
7. http://www.crosstalkonline.org/storage/issue-archives/2009/200903/200903-0-Issue.pdf
8. http://www.crosstalkonline.org/storage/issue-archives/2009/200909/200909-0-Issue.pdf
9. http://www.crosstalkonline.org/storage/issue-archives/2010/201003/201003-0-Issue.pdf
10. http://www.crosstalkonline.org/storage/issue-archives/2010/201009/201009-0-Issue.pdf
11. http://www.crosstalkonline.org/storage/issue-archives/2009/200909/200909-0-Issue.pdf
12. http://www.crosstalkonline.org/storage/issue-archives/2010/201003/201003-0-Issue.pdf
13. http://www.crosstalkonline.org/storage/issue-archives/2010/201009/201009-0-Issue.pdf

		of us refer to it. The absence of a planned approach opens the possibility for application flaws which adversaries could potentially exploit. Exploiting application flaws is a likely scenario, considering a Gartner report that states that 75 percent of successful attacks are targeted towards vulnerabilities at the application layer.
Being Explicit About Security Weaknesses	3/1/07	Robert A. Martin. Software acquirers want assurance that the products they are obtaining are reviewed for known types of security weaknesses. Acquisition groups in large government and private organizations are beginning to use such reviews as part of future contracts, but the tools and services for performing them are new and, until recently, there was no nomenclature, taxonomy, or standard to define their capabilities and coverage. A standard dictionary of software security weaknesses has been created by the community to serve as a unifying language of discourse and as a measuring stick for comparing tools and services.
Building Secure Systems Using Model-Based Engineering and Architectural Models	9/1/08	Dr. Jörgen Hansson, Dr. Peter H. Feiler, and John Morley. The Department of Defense's policy of multi-level security (MLS) has long employed the Bell-LaPadula and Biba approaches for confidentiality and integrity; more recently, the multiple independent levels of security/ safety (MILS) approach has been proposed. These approaches allow designers of software-intensive systems to specify security levels and requirements for access to protected data, but they do not enable them to predict runtime behavior. In this article, model-based engineering (MBE) and architectural modeling are shown to be a platform for multi-dimensional, multi-fidelity analysis that is conducive for use

		with Bell-LaPadula, Biba, and MILS approaches, and enables a system designer to exercise various architectural design options for confidentiality and data integrity prior to system realization. In that way, MBE and architectural modeling can be efficiently used to validate the security of system architectures and, thus, gain confidence in the system design.
Considering Software Supply Chain Risks©	9/1/10	Dr. Robert J. Ellison and Dr. Carol Woody. As outsourcing and commercial product use increase, supply chain risk becomes a growing concern for software acquisitions. Hardware supply chain risks include manufacturing and delivery disruptions and the substitution of counterfeit or substandard components. Software supply chain risks, usually during development, include third-party product tampering or the introduction of exploitable software defects. This article identifies several current practices that can be incorporated in an acquisition to reduce those risks.
Enhancing the Development Life Cycle To Produce Secure Software	9/1/08	Karen Mercedes Goertzel. Over the past decades, efforts to enhance software development life cycle (SDLC) practices have been shown to improve software quality, reliability, and fault-tolerance. More recently, similar strategies to improve the security of software in organizations such as Microsoft, Oracle, and Motorola have resulted in software products with less vulnerabilities and greater dependability, trustworthiness, and resilience. In its mission to improve the security of software used in America's critical infrastructure and information systems, the Department of Homeland Security's (DHS) Software Assurance Program has sponsored the creation of the book Enhancing the Development

		Life Cycle to Produce Secure Software, a source of practical information intended to help developers, integrators, and testers identify and systematically apply security and assurance principles, methodologies, and techniques to current SDLC practices, and thereby increase the security of the software that results. Unlike the numerous other books on secure software development, Enhancing the Development Life Cycle does not espouse any specific methodology, process model, or development philosophy. Instead it explains the essentials of what makes software secure, and takes an unbiased look at the numerous security principles and secure development methodologies, practices, techniques, and tools that developers are finding effective for developing secure software – information that readers can leverage in defining their own SDLC security-enhancement strategies.
Gaining Software Assurance through the Common Criteria	9/1/10	Patti Spicer. The Common Criteria (ISO/IEC 15408) is often misunderstood in terms of its scope and purpose: It does not provide a guarantee of a product's security posture, but does provide assurance of its operation. This article[14] seeks to provide background information about the Common Criteria (CC), and a better understanding of the software assurance that CC product certification truly provides. Go to the article[15]
High-Leverage Techniques for Software Security	3/1/07	Idongesit Mkpong-Ruffin and Dr. David Umphress. Software security addresses the degree to which software can be exploited or misused. Software development approaches tend to polarize security efforts as being reactive or proactive; a blend of both approaches is needed in practice.

		Three categories of tools provide such a blend: threat modeling, risk analysis, and security assessment and testing. These tools provide leverage as they are currently in use as quality assurance methods and can be modified with relatively little effort to address security.
Information Assurance Applications in Software Engineering Projects	9/1/10	Lt. Col. Thomas A. Augustine (Ret.) and Dr. Lori L. DeLooze. Four recent capstone projects by students in the U.S. Naval Academy's (USNA) Department of Computer Science offer some interesting insights into methodologies for information assurance (IA). This article looks at the tasks and challenges of each project and consolidates the experiences into lessons learned for designing and implementing software or systems that incorporate the IA concepts of confidentiality, data integrity, authentication, and system availability.
Practical Defense in Depth	9/1/08	Michael Howard. As part of its ongoing commitment to Bill Gates' vision of Trustworthy Computing, Microsoft officially adopted important security- and privacy-related disciplines to its software development process. These changes, called the Security Development Lifecycle (SDL) have led to a demonstrable reduction in security vulnerabilities in products such as Microsoft's Windows Vista operating system and its SQL Server 2005 database. The purpose of this article is not to describe the SDL in detail, but to outline some of the practical defensive measurements in use at Microsoft required by the SDL. If Microsoft's SDL is new to you, refer to the page 16 sidebar, "A Brief SDL Overview."
Secure Coding Standards	3/1/07	James W. Moore and Robert C. Seacord. Inherent weaknesses in

		programming languages contribute to software vulnerabilities. Increasingly, organizations are producing standards to improve software security. Current efforts to develop software security standards are surveyed, and two such efforts are described in detail. An international standards group is writing a document providing guidance to users of programming languages on how to avoid the vulnerabilities that exist in the programming language selected for a particular project. Carnegie Mellon University's (CMU's) Computer Emergency Response Team (CERT) is developing secure coding practices for the C and C++ programming languages.
Software Assurance: Five Essential Considerations for Acquisition Officials	5/1/07	Mary Linda Polydys and Stan Wisseman. Software Assurance (SwA) is a key element of national security; it is critical because dramatic increases in business and mission risks are attributable to exploitable software. A recent Chief Information Office (CIO) Executive Council poll indicated that the top two most important attributes of software are reliable software that functions as promised and software free from security vulnerabilities and malicious code. The acquisition process can be leveraged to achieve these important attributes. As part of the Department of Homeland Security (DHS) and Department of Defense (DoD) SwA initiative, a working group developed a guide, Software Assurance in Acquisition: Mitigating Risks to the Enterprise <https://buildsecurityin.us-cert.gov[16]>, for acquisition officials on how to incorporate SwA considerations in key decisions throughout the acquisition process.
Software Assurance Practice at Ford: A Case Study	3/1/09	Dr. Nancy R. Mead, Dr. Dan Shoemaker, and Jeffrey A. Ingalsbe. Software pervades our

		technological society, handling our financial transactions, managing power transmission, facilitating most forms of communication, and keeping us safe. This makes defects in software one of the most potent threats to our national security, and turns identification of best practices in software development, acquisition, and long-term use the highest national priority. This article presents the best practices employed by the Ford Motor Company to develop and maintain their software assets.
Static Analysis Is Not Just for Finding Bugs	9/1/10	Dr. Yannick Moy. Static analysis tools are gaining popularity for safeguarding against the most common causes of errors in software. The main focus of these tools is on automatic bug-finding—the first stage in a two-phase process where the tool finds bugs and the human then corrects them. This article explains that such a goal is too narrow for critical software assurance (SwA). Instead, static analysis tools should adopt a broader perspective: computing properties of software.
Static Analyzers in Software Engineering	3/1/09	Dr. Paul E. Black. Static analyzers can report possible problems in code and help reinforce the good practices of developers. This article contrasts the strengths of static analyzers with testing and discusses the current state-of-the-art.
Studying Software Vulnerabilities	9/1/10	Dr. Robin A. Gandhi, Dr. Harvey Siy, and Yan Wu. There have been several research efforts to enumerate and categorize software weaknesses that lead to vulnerabilities. To consolidate these efforts, the Common Weakness Enumeration (CWE) is a community-developed dictionary of software weakness types and their relationships. Yet, using the CWE to study and prevent vulnerabilities in specific software

		projects is difficult. This article presents a novel approach for using the CWE to organize and integrate the vulnerability information recorded in large project repositories.
The Balance of Secure Development and Secure Operations in the Software Security Equation	9/1/10	Sean Barnum. Software security is about reducing the risk that software poses to those who use it or are affected by it. This requires thought and action more than simply at the point of development or use. It requires a more holistic approach, balancing secure development and secure operations. The bad news is that these two capable domains typically do not interact much or understand each other. The good news is that there are active ongoing efforts focused on addressing this gap.
The Security of Web Services as Software	9/1/07	Karen Mercedes Goertzel. To help creators of Web services and Service-Oriented Architectures (SOAs) understand and address the security challenges that confront them, the National Institute of Standards and Technology (NIST) is getting ready to publish a new Special Publication (SP) 800-95, Guide to Secure Web Services. This SP describes Web service security standards and explains how to develop Web services and SOA portals using technologies based on those standards. However, neither SP 800-95 nor the standards it describes address a critical challenge: the security of Web services as software. Without considering software security, developers cannot create Web services that are truly trustworthy. This article describes both the content of SP 800-95 and highlights its critical omissions in terms of measures needed to produce Web service software that is in and of itself secure.

The Software Quality Challenge	6/1/08	Watts S. Humphrey. Many aspects of our lives are governed by large, complex systems with increasingly complex software, and the safety, security, and reliability of these systems has become a major concern. As the software in today's systems grows larger, it has more defects, and these defects adversely affect the safety, security, and reliability of the systems. This article explains why the common test-and-fix software quality strategy is no longer adequate, and characterizes the properties of the quality strategy we must pursue to solve the software quality problem in the future.
The Use and Limitations of Static-Analysis Tools to Improve Software Quality	6/1/08	Dr. Paul Anderson. Advanced static-analysis tools have been found to be effective at finding defects that jeopardize system safety and security. This article describes how these work and outlines their limitations. They are best used in combination with traditional dynamic testing techniques, and can even reduce the cost to create and manage test cases for stringent run-time coverage.
Two Initiatives for Disseminating Software Assurance Knowledge	9/1/10	Dr. Nancy R. Mead and Dr. Dan Shoemaker. Education in software assurance (SwA) is an essential element in the effort to produce secure code. This article describes two efforts that support national cybersecurity education goals: development of SwA learning artifacts that can be integrated into conventional learning environments and establishment of a reference curriculum for a master's degree program, known as the MSwA.